Large-Scale Distributed Systems

Panel Presentation, SDP Workshop

Priya Narasimhan
Institute of Software Research International
School of Computer Science
Carnegie-Mellon University

priya@cs.cmu.edu
http://www.cs.cmu.edu/~priya

Carnegie Mellon

Scaling Middleware Systems

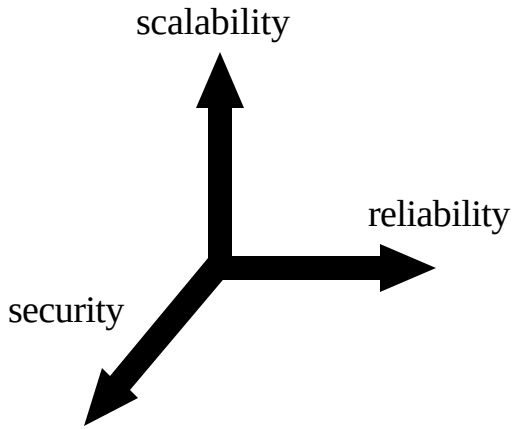
- **What does large-scale mean?**
  - Number of objects/components increases
  - Number of operations/sec increases
  - Number of nodes increases
  - Distance between nodes increases
  - Number of administrators increases

- **How do we "grow" middleware gracefully without degrading any of its services?**

© 2001 Priya Narasimhan

Marrying/Composing "-ilities"

- **System properties or "-ilities"**
  - Reliability, security, real-time, ….
  - And, of course, scalability!
  - Multi-dimensional property space



- **Marrying various "-ilities"**
  - How do they impact each other?
  - Trade-offs, compromises, conflicts in the marriage
  - How is the marriage impacted when the number of clients/objects/nodes/operations/etc. increases?
  - How is the marriage impacted in the presence of resource constraints?

© 2001 Priya Narasimhan

Challenges

- **Defining Metrics**
  - Quantifying an "-ility" and the composition of "-ilities"
  - Developing benchmarks to evaluate "-ilities"
- **Performing Trade-off Analysis**
  - Analyzing the marriage of "-ilities" in a resource-aware manner
  - Embodying the results in techniques for self-adaptive systems
- **Providing Transparency**
  - Hiding the intricacies of the "-ilities" within the infrastructure so that the application logic/programming is simple
- **Developing Tunability APIs**
  - Deciding how much control the user can have
  - Finding and standardizing APIs that are best for tunability

© 2001 Priya Narasimhan

Long-Term Benefits

- **Metrics**
  - Objective evaluation/comparison of middleware and "-ilities"
- **Trade-off Analysis**
  - Middleware that can support multiple "-ilities" simultaneously
  - Trade-off-aware middleware that can adapt to changing resources and application requirements, and be sensitive to resource constraints
- **Transparency**
  - Application programmers do not need training in the "-ility"
  - Savings in cost, development time, maintenance
- **Tunability APIs**
  - Customization of a system to meet specific needs
  - Prevents illegal composition of "-ilities" by inexpert users

© 2001 Priya Narasimhan